**Breve Descripción del Problema.**

Se va a realizar un video juego llamado “Space Invaders” donde el jugador moviendo su nave principal horizontalmente en la parte inferior de la pantalla deba disparar a los enemigos que se estarán acercando poco a poco, esto con el fin de que los enemigos no lleguen donde se encuentra la nave. Existirán distintos tipos de hileras de enemigos así como jefes, donde el jugador deberá ir sobrepasando cada uno haciendo que estos nunca lleguen a la parte inferior.

**Historias de Usuario:**

Como jugador quiero:

1. Que la nave se mueva horizontalmente en la parte inferior de la pantalla.
2. Que hayan enemigos en la parte superior de la pantalla.
3. Que los enemigos estén acomodados por hileras los cuáles se muevan hacia abajo y hacia los lados.
4. Que la nave deba eliminar los enemigos mediante disparos
5. Que la nave deba evitar que los enemigos lleguen a la parte inferior de la pantalla.
6. Que existan varios tipos de hileras.
7. Que la velocidad de los enemigos aumente conforme se avanza de nivel.
8. Que cada vez que se destruya un enemigo, estos se desplacen hacia el centro.
9. Que todos los enemigos sean iguales.
10. Que haya un jefe dependiendo el tipo de hilera, para destruir a este se requieren entre 2 y 5 disparos.
11. Que cuando se destruya el jefe se pase al siguiente nivel u otro tome su lugar (dependiendo de la hilera)
12. En la pantalla se deberá indicar el tipo de hilera actual y el próximo por salir, el nivel actual y el puntaje actual.
13. Que en un tipo de hilera los enemigos tengan distintas resistencias.

**Historias de Usuario según Criticalidad y Secuencia de Uso**

**Orden por secuencia de uso:**

1, 2, 4, 3, 5, 8, 9, 6, 10, 11, 7, 13

A continuación se mostrará una matriz para asignar la criticalidad y secuencia de uso que van a tener las tareas establecidas en cada bloque de historias de uso. La clasificación de los colores para la matriz se explicará a continuación:

1. Color Rojo: representa el nivel de prioridad “Alto”, por lo que debe realizarse con mayor urgencia, y se debe considerar muy importante para el desarrollo del proyecto.
2. Color Amarillo: representa el nivel de prioridad “Medio”, por lo que es poco urgente pero importante.
3. Color Verde: representa el nivel de prioridad “Bajo”, por lo que es poco urgente y no tan importante.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Secuencia de Uso | | | |
| Criticalidad |  |  |  |  |
| 1, 2, 4 | 3, 5 |  |
|
| ↑ Minimal System Span ↑ | | | |
|  |  | 6, 8, 10 | 11 |
|
|  |  | 9 | 7, 13 |
|

Tabla 1. Matriz de Criticalidad y Secuencia de Uso de las Historias de Usuario

**Plan de Iteraciones**

**Cronograma**

Bloque 1.

- Búsqueda de los sprites y el escenario para todos los elementos gráficos del juego.

- Hacer que la nave principal aparezca sobre el escenario.

- Hacer que la nave se mueva horizontalmente.

Bloque 2.

- Hacer que los enemigos aparezcan en la parte superior.

- Hacer que los enemigos tenga sus sprites en movimiento.

- Hacer que los enemigos se muevan hacía los lados y hacía abajo.

**public class** nodo<T> {  
  
 **private** T **dato**;  
 **private** nodo<T> **siguiente**;  
  
 **public** nodo(){  
 **this**.**dato** = **null**;  
 **this**.**siguiente** = **null**;  
 }  
  
 **public** T getDato() {  
 **return dato**;  
 }  
  
 **public void** setDato(T dato) {  
 **this**.**dato** = dato;  
 }  
  
 **public** nodo<T> getSiguiente() {  
 **return siguiente**;  
 }  
  
 **public void** setSiguiente(nodo<T> siguiente) {  
 **this**.**siguiente** = siguiente;  
 }  
}

**import** javax.print.DocFlavor;  
  
**public class** ListaSimple<T> {  
  
 **private** nodo primero;  
 **private int** tamaño;  
  
 **public void** ListaSimple(){  
 primero = **null**;  
 }  
  
 **public boolean** isEmpty(){  
 **return** primero == **null**;  
 }  
  
 **public int** getTamaño(){  
 **return** tamaño;  
 }  
  
 **public void** agregarAlFinal(T valor){  
 nodo<T> nuevo = **new** nodo<T>();  
 nuevo.setDato(valor);  
  
 **if**(**this**.isEmpty()){  
 **this**.primero = nuevo;  
 }  
 **else**{  
 nodo<T> aux = **new** nodo<T>();  
 aux = primero;  
 **while** (aux.getSiguiente() != **null**){  
 aux = aux.getSiguiente();  
 }  
 aux.setSiguiente(nuevo);  
 }  
 tamaño += 1;  
 }  
  
 **public void** agregarAlInicio(T dato) {  
 nodo<T> nuevo = **new** nodo<T>();  
 nuevo.setDato(dato);  
  
 **if** (**this**.isEmpty()) {  
 **this**.primero = nuevo;  
 } **else** {  
 nuevo.setSiguiente(**this**.primero);  
 **this**.primero = nuevo;  
 }  
 tamaño += 1;  
 }  
  
 **public** nodo getPrimero(){  
 **return this**.primero;  
 }  
  
 @Override  
 **public** String toString() {  
 String lista = **new** String();  
 nodo aux = **new** nodo();  
 aux = primero;  
 **for**(**int** i = 0; i<**this**.tamaño; i ++){  
 lista += aux.getDato() + **", "**;  
 aux = aux.getSiguiente();  
 }  
 **return** lista;  
 }  
}

**public class** main {  
  
 **public static void** main(String[] args){  
 ListaSimple lista = **new** ListaSimple();  
  
 lista.agregarAlInicio(**"hola"**);  
 System.***out***.println(lista);  
 lista.agregarAlInicio(**"hola1"**);  
 System.***out***.println(lista);  
 lista.agregarAlInicio(**"hola2"**);  
 System.***out***.println(lista);  
 lista.agregarAlFinal(**"adios"**);  
 System.***out***.println(lista);  
  
  
 }  
}